BSTZ No. 42390P11478
Express Mail No. EL651844095US

UNITED STATES PATENT APPLICATION

FOR

SYSTEM AND METHOD FOR EXTERNAL BUS DEVICE SUPPORT

Inventors:

Rajeev K. Nalawadi
Frederick H. Bolay

Prepared by:

Blakely, Sokoloff, Taylor & Zafman LLP
12400 Wilshire Boulevard, Suite 700
Los Angeles, California 90025
(310) 207-3800

SYSTEM AND METHOD FOR EXTERNAL BUS DEVICE SUPPORT

Field of the Invention

[0001] This invention relates to computers and computing devices that include external bus enabled devices, such as Universal Serial Bus (USB) devices, and more particularly to providing support for USB and other external bus enabled devices during system boot up, before operating system support for USB and other external bus enabled devices is present, by issuing a periodic software interrupt.

Background

[0002] As personal computers have evolved, the processing power, available memory, available peripheral devices and personal computer features such as the kinds of peripheral connection methods have increased with each passing year. When powering on a personal computer, a basic input-output system (BIOS) controls what occurs. A BIOS may check the status of various hardware components and devices and receive input from the hardware devices, such as keyboard, mouse, and other input devices. To receive such input, the BIOS must be able to receive data from and otherwise communicate with the devices attached to or included in the system. For example, it may be necessary for the BIOS to receive user input from a keyboard or for the BIOS to initiate execution of startup software to be read from, for example, a floppy disk, hard disk, and compact disk read-only memory (CD-ROM). To provide support for all of these devices, the BIOS contains software that is executed in system memory that serves as device drivers or other software that supports various devices until an operating system including drivers for the particular devices is loaded.

BRIEF DESCRIPTION OF THE DRAWINGS

[0003] The invention described herein is illustrated by way of example and not by way of limitation in the figures of the accompanying drawings in which like references indicate similar elements. It should be noted that references to "an" or "one" embodiment in this disclosure are not necessarily to the same embodiment, and such references mean at least one.

[0004] **Figure 1** illustrates a hardware environment in which one embodiment of the invention executes.

[0005] **Figure 2A** illustrates a diagram of Universal Serial Bus (USB) controller registers and system memory showing the organization of USB data.

[0006] **Figure 2B** illustrates a diagram of system memory showing the location of Universal USB support memory.

[0007] **Figure 3** illustrates a flow of actions taken pursuant to one embodiment of the invention.

DETAILED DESCRIPTION

[0008] In many current computers, support must be provided for external devices while the system is booting up and while the operating system is loading, until the operating system and device drivers have completed loading. To allow the system to support one or more external peripheral devices during system boot up, the basic input-output system (BIOS) includes support software to process input from external devices during system boot up.

[0009] In many current computer systems, and particularly processors available from Intel Corporation having a 32-bit architecture known as IA-32 and other processors compatible with the 32-bit Intel Architecture, (See IA-32 Intel Architecture Software Developer's Manual available from Intel Corporation, Santa Clara, California.), before operating system support for external devices, such as, Universal Serial Bus (USB) devices is present, the BIOS enables the chipset (hardware) to convert all enabled USB hardware interrupt occurrences into USB legacy system management interrupts (SMIs). The SMI that is generated internally by the chipset (hardware) after conversion of the USB interrupt to an SMI is referred to herein as a "USB legacy SMI". There is a single SMI pin connection from the chipset (hardware) to the processor, and there are various individual sources of SMIs in the system. The chipset (hardware) acts as a collector for all SMI events in the system. When any of the SMI sources is active in the system, the chipset (hardware) asserts the SMI pin to the processor. This SMI pin assertion from the chipset (hardware) to the processor is referred to herein as a "Hardware SMI". When the SMI signal is asserted, the processor enters System Management Mode (SMM) and the BIOS code executes. The

BIOS includes SMM software that checks for various registers in the chipset (hardware) to determine the source of the SMI and handle it appropriately. Among the various SMI sources in the system, there is a capability provided in the chipset (hardware) to generate an SMI periodically based on expiration of an internal hardware timer. The granularity and the occurrence of this periodic SMI is programmable by the BIOS. This SMI is referred to herein as a "periodic software SMI". A periodic software SMI issues, is generated, whenever an internal timer in the chipset times-out. The periodic software SMI can occur in conjunction with SMIs, generated by other sources, such as by software writing to known specific registers in the chipset (hardware). Hence, the periodic software SMI may be referred to as an "asynchronous periodic software SMI". The granularity of the periodic software SMI occurrence may vary among various chipsets that may be used in a system, but a few feasible granularities are provided in all the chipsets. The BIOS SMM software is usually capable of handling multiple SMI occurrences during one instance of the SMM code execution. In addition, the chipset also provides various software controls to enable and disable the various USB interrupt sources, and also to enable and disable the occurrence of USB legacy SMIs. However, many existing computer systems do not have chipset (hardware) that allows for the conversion of USB hardware interrupts into legacy USB software interrupts.

[0010] In the system and method described herein, when the chipset (hardware) does not have the capability to generate a USB legacy SMI, software added to a traditional BIOS sets an asynchronous periodic software SMI, and parses through and processes completed USB descriptors of one or more USB host controller(s) when the periodic software SMI is generated. The chipset (hardware) provides different time periods for the occurrence of the asynchronous periodic software SMI generation. On every asynchronous periodic software SMI occurrence, the USB support software parses through completed USB descriptors. In one embodiment, the USB support software dynamically increases or decreases the periodic SMI occurrence to a faster or slower rate to handle the data transfer rate of attached USB devices. To parse through the USB descriptors without interference from existing components, the USB support software disables the conversion of USB hardware interrupts into USB legacy software interrupts and also disables the chipset (hardware) from generating USB interrupts.

[0011] **Figure 1** illustrates a hardware environment in which one embodiment of the invention executes. A computing device such as personal computer 100 may include processor 102, memory 104, storage device 106, and communications device 108 coupled to bus 140. In one embodiment, the processor may be a processor conforming to or compatible with the 32-bit Intel Architecture known as IA-32. In one embodiment, memory 104 may be any kind of random access memory (RAM). In one embodiment, storage device 106 may be any kind of machine readable medium including, for example, magnetic media such as disk drives and magnetic tape; optical drives such as compact disk read only memory (CD-ROM) and readable and writeable compact disks (CD-RW); stick and card memory devices; ROM, RAM, flash memory devices and the like; whether internal, such as storage device 106, directly coupled such as such as external device 126, accessible locally or remotely via a network, and via electrical, optical, acoustical or other form of propagated signals (*e.g.*, carrier waves, infrared signals, digital signals, etc.) via communications device 108. In one embodiment, communications device 108 may be a modem, network interface unit, or other communications device that allows for communication with other computing devices.

[0012] Personal computer 100 may include graphics adapter 110 that allows for the display of information such as text and graphics on display monitor 112. Graphics adapter 110 may provide support for the video graphics array (VGA), super VGA (SVGA) and/or other graphics standards. In one embodiment, graphics adapter 110 may be included as part of an integrated processor/ graphics chipset. In one embodiment, graphics adapter 110 may be coupled to the system via an Accelerated Graphics Port (AGP) interface supported by the chipset used in personal computer 100.

[0013] Personal computer 100 may include an external bus controller, such as, USB host controller 120 which allows for two-way communication with external USB devices, such as external device 126. Although only one USB host controller is shown, multiple USB host controllers and/or other external bus controllers may be physically included in the personal computer. In various embodiments external device 126 may be any USB enabled device, such as, for example, a machine readable medium reader and/or writer, a digital camera, a printer, a digital music player/recorder such as an MP3 player, etc. Various USB enabled input devices may also be coupled to personal

computer 100 via USB controller 120, such as, for example, keyboard 122 and mouse 124. In one embodiment, a USB enabled biometric device such as fingerprint reader 128, retinal scanner or voice recognition device may be coupled to personal computer 100 via USB controller 120. Although these devices are referred to and/or are depicted as external devices or external bus enabled devices, the devices may exist inside or otherwise within the enclosure of a computer or other computing device. These devices are called external devices because they communicate via what are well known as external bus protocols via external bus controllers, such as USB. In addition, each of the devices may include its own controller, such as a USB host controller. In one embodiment, the USB devices and host controller(s) may conform to the Universal Serial Bus Specification, Revision 1.1, dated September 23, 1998 and/or Revision 2.0 dated April 29, 2000 (the "USB Specification").

[0014] Graphics adapter 110 and USB controller 120 are each coupled to bus 140. Although only one each of processor 102, external device 126 and storage device 106 are depicted, multiple processors and multiple storage devices may be included in personal computer 100, and multiple external USB devices may be coupled to personal computer 100 via USB controller 120. A basic input-output system (BIOS) 130 that may include a USB support component such as USB support software 132 is also coupled to bus 140. BIOS 130 may be software stored in hardware, such as, for example, an electrically erasable programmable read-only memory (EEPROM) device, a flash memory device, etc. In one embodiment, the BIOS and/or the USB support software may be updated or installed from computer instructions stored on a machine readable medium such as a floppy disk or CD-ROM. In this embodiment, the instructions are copied or otherwise transferred from the machine readable medium to the BIOS chip such that they are executed during all subsequent booting up of the system. In one embodiment, bus 140 may be a USB such that all components and devices communicate via USB. USB support software 132 is discussed in more detail below.

[0015] According to one embodiment of the system and method presented herein, during start-up of the computing device, the computing device may request input from the user or the user may access an input device, such as a keyboard, to alter the regular boot process. To request and receive user input during boot up, software in

the BIOS may display information on a display to the user, and the user may provide input via a keyboard or keypad, a mouse, a biometric device, and/or another input device, each of which may be coupled to the computing device via USB. In addition, in one embodiment, a user identification card may be presented via a USB enabled card reader (not shown) to authenticate the user's access to the computing device. In some situations, a software update or other software may need to be read or otherwise executed by the BIOS from a storage device coupled to the computing device via the USB controller. To support the multitude of devices which may be coupled to the computing device via one or more USB host controllers, the BIOS stores device information and/or device driver information in the computing device's memory, including USB device data such as transfer descriptors.

[0016] **Figure 2A** illustrates a diagram of USB controller registers and system memory showing the organization of USB data. USB host controllers have 1024 frames associated with them in memory such that each USB host controller is represented in memory by a 1024 x 4 byte region. (See the USB Specification.). In one embodiment, a portion of system memory is obtained to be used to be used by USB support software to maintain USB data. USB support memory 200 may include a portion of memory designated USB host controller memory 220 to hold USB host controller data.

[0017] USB host controllers include well-known registers that are used to access and control the USB host controller and the controllers communication with USB devices. In one embodiment, a USB host controller may include registers 210. Registers 210 include a register or registers that points to frames in USB host controller data 220, such as frame base 212 and frame number 214. Frame base 212 and frame number 214 point to a base address of a list of frames and a frame number, or offset, in USB host controller memory 220. Each frame 222 in USB host controller memory 220 points to a transfer descriptor 230 having USB data 240 associated with the transfer descriptor. The contents of frame 222 is a pointer to a memory address where the TD resides. In one embodiment, a USB schedule for each USB host controller consists of 1024 frames. Each frame 222 may point to one or more USB descriptors. In a classic USB host controller implementation, the USB host controller takes one millisecond to execute each frame. If all the USB descriptors associated with the frame execute

within a one millisecond period, the hardware waits for the completion of one millisecond to start executing the next frame to ensure that the 1024 frames have completed execution.

[0018] **Figure 2B** illustrates a diagram of system memory showing the location of Universal USB support memory. In one embodiment, the system memory may be mapped according to a predetermined specification, such as, for example, the memory structure defined in the Advanced Configuration and Power Interface (ACPI) specification (rev. 2.0 dated July 27, 2000; see also ACPI Component Architecture Programmer Reference, rev. 1.05 dated February 27, 2001 available from Intel Corporation, Santa Clara, California). In one embodiment, a portion of the non-volatile sleeping (NVS) region of memory may be used by the USB support software of the BIOS to provide USB device support during system boot up until an operating system including USB device drivers has completed loading.

[0019] Pursuant to the ACPI specification, the system memory is mapped according to memory structure 250. Memory structure 250 includes compatibility memory 252, located in the region of memory located at from 0 to 640 Kbytes. Compatibility holes 254 are located at from 640 Kbyte to 1 Mbyte. In traditional systems, the BIOS was limited to accessing compatibility memory 252 and compatibility holes 254. The compatibility memory 252 and compatibility holes 254 may be referred to as the programmable address map (PAM) region of memory 256. The ACPI specification also defines operating system usable system memory as contiguous RAM 258 which is located at from 1 Mbyte to the bottom of memory one 260. In one embodiment, bottom of memory one 260 may be at 8 Mbytes. ACPI tables 262 are located in the region from bottom of memory one 260 to an area referred to in the ACPI specification as the top of memory one, denoted 264. The area of memory between the top of memory one and what the ACPI specification refers to as the top of memory two, denoted 272, is defined to include two regions of memory, ACPI NVS memory 266 and ACPI reserved memory 270. It is a portion of ACPI NVS memory 266 that the method and system described herein uses, in one embodiment, to provide USB support. One portion of memory is denoted USB support memory 268 within or part of ACPI NVS memory 266. It is this portion of system memory that is used for the data format described above regarding **Figure 2A**. To complete the

memory structure description, the ACPI specification also defines a "no memory region" referred to in one embodiment as Peripheral Component Interconnect (PCI) bus addressable memory 280 located between top of memory two and boot base 292. The memory area from boot base 292 to the top of memory 294, at, in one embodiment, 4 Gbytes, is virtually used for boot ROM 290.

[0020] **Figure 3** illustrates a flow of actions taken pursuant to one embodiment of the invention. When a system such as a computing device is powered on, the boot process begins, as shown in block 310. The BIOS code executes, as shown in block 312. This process as a whole is sometimes referred to as power on self test (POST). The system memory initializes, as shown in block 314. External bus support software such as USB support software then obtains a portion of memory and constructs USB device data structures, as shown in block 316. The USB device data includes the data described above regarding **Figure 2A**. The USB support software then sets the system hardware to generate a periodic SMI at a slow rate, and disables the system hardware from generating USB legacy SMIs, as shown in block 318. There are various periodic SMI occurrence rates that can be programmed in the chipset (hardware). Periodic SMI rates available on many systems are every 64 seconds (approximately one minute), 32 seconds, 16 seconds, 8 seconds, 64 milliseconds, 32 milliseconds, 16 milliseconds, 8 milliseconds, and one millisecond. These periodic SMI rates are provided as an example, and some hardware embodiments may provide a much wider range of periodic occurrences that may be used by the USB support software and the BIOS. The SMI is initially set by the USB support software to be generated at a slow rate, such as, for example, every minute or every 30 seconds, so as not to unnecessarily impede performance of the system during the boot process by causing unneeded processing of SMIs. The USB support software then starts the USB host controller such that the USB host controller processes USB device descriptors using the obtained portion of memory, the USB support memory, as shown in block 320. To provide USB device support during the remainder of the boot process, execution continues in parallel at block 330 and at block 350.

[0021] The USB host controller processes USB device data by transmitting data to USB devices and/or receiving data from USB devices, as shown in block 330. The system hardware then generates a periodic SMI, as shown in block 332. The USB

support software then checks for the presence of USB device drivers, as shown in block 334. The USB frame list base address register is set to reflect the address of the location of the USB support memory until the device drivers are loaded. Because the USB frame list base address register is modified when the operating system has loaded the USB drivers, the USB support software uses the technique of looking at the USB frame list base address to determine whether the operating system drivers are loaded (see frame base 202 of **Figure 2A**). The USB frame list base address register is located in the USB host controller chipset (hardware) and contains a pointer to the start of the USB frames, which may also be referred to as the USB schedule, constructed by the USB support software or operating system device drivers. When the USB host controller is "started" or "activated" by the USB support software or the operating system drivers, the USB host controller chipset (hardware) executes the USB descriptors located in memory pointed to by the current contents of the USB frame list base address register. The contents of the USB frame list base address register are incremented by a value of four (4) automatically by the USB host controller chipset (hardware) after execution. The USB frame list base address register contents keep rolling in a round-robin fashion, thus executing the active USB descriptors until the USB host controller is "stopped".

[0022] If USB device drivers are absent, flow continues at block 334. The USB support software then parses through USB descriptors and processes USB data, as shown in block 336. The parsing involves traversing the descriptors and other data described above regarding **Figure 2A**. When the periodic software SMI occurs, after parsing, the processing performed by the USB support software may include passing the USB data to the processor, a keyboard controller, a mouse controller, etc. In this way, the USB data is transferred to the BIOS code or other boot software that is executing. The USB support software may then adjust the rate of the periodic SMI based on the USB data traffic, as shown in block 338. More specifically, in one embodiment, the USB support software may change the periodic software SMI occurrence rate to dynamically complement the rate of the USB data. That is, the SMI occurrence period may be increased, sped up, or decreased, slowed down, as needed. Such speed or period adjustments are often needed when the USB descriptors are transferring user based input. When no input data is received, the rate is reduced; when

input data is received, the rate is increased and remains at a faster level, such as, for example, every 16 milliseconds or every 8 milliseconds, until the amount of received data drops off. Execution then continues at block 330.

[0023] After the check for the presence of USB device drivers is made, as shown in block 334, if USB device drivers are present, flow continues a block 340, and the USB support software de-allocates the obtained portion of memory, the USB support memory, disables the periodic SMI, and exits, as shown in block 340. USB support is thereafter provided by device drivers, as shown in block 342. In one embodiment, the USB support software may at some point regain control from the operating system and then, again, begin issuing periodic software SMIs and processing USB data. One of example of when this may occur is during a soft reboot.

[0024] In parallel with the flow beginning at block 330 is the flow beginning at block 350. The BIOS code continues to execute, as shown in block 350. At some point, the BIOS code passes control to the operating system, as shown in block 352. The operating system loads, including loading USB (and other) device drivers while user input offered during operating system and device driver loading and/or configuration is processed by the USB support software responsive to the periodic SMI, as shown in block 354. The USB support software generates periodic SMIs to invoke the processing of USB data during loading to the operating system. If the USB support software is not included in the system to generate periodic SMIs to invoke the processing of USB data during loading of the operating system, the operating system may hang during boot or input data may be lost when control of the USB devices and USB host controller transitions from the BIOS to the operating system device drivers. The USB support software that causes the generation of periodic SMIs to invoke the processing of USB data during loading of the operating system alleviates missed USB input and alleviates the system hanging during operating system boot should a USB interrupt not be handled.

[0025] In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes can be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification

and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.